Modeling Sleep Time for Query-Based Wireless Sensor Networks

Deepak Jha*, Kai Howelmeyer†, Faisal Nawab*
* Dept. of Computer Science, University of California at Santa Barbara, Santa Barbara, CA, {nawab, deepakjha}@cs.ucsb.edu
† University of California at Santa Barbara, Santa Barbara, CA, hoewelmeyer@umail.ucsb.edu

***Abstract*—This paper consider query-based wireless sensor networks (WSN). Nodes are requested for information on demand. A design of such a network is presented and a prototype implementation is deployed. We study energy efficiency in such systems. We found that even while having the same cumulative sleep duration, the duration of each sleep cycle have an effect on energy consumption. This is due to overhead caused by polling at the end of each sleep cycle. Thus, we tackle the problem of maximizing sleep periods while ensuring a QoS requirement, namely delay. We explore a model that will enable us to calculate suitable sleeping times. We perform a simulation study and an experimental validation of our findings.**

***Index Terms*—Wireless Sensor Networks (WSN), Energy Efficiency, ZigBee, Sleeping, Polling.**

I. Introduction

Wireless Sensor Networks (WSNs) use sensors to measure a wide variety of live data and actuators to manipulate their environments. Data is communicated between nodes and sinks. WSNs are self-configuring, inherently small, low-cost, and low-power. This makes it a suitable technology for use in monitoring and tracking applications. Many of the envisioned applications of sensor networks require a large-scale deployment that comprises thousands of sensor nodes, e.g., the Internet of things [7] and sensor database systems [8]. With a large number of sensor nodes it becomes challenging to maintain all collected data in a centralized unit (or multiple centralized units). Moreover, the demand for data might vary from node to node. Keeping the node awake at all times is a waste of communication resources and will ultimately drain the nodes power prematurely. An energy-aware demand-driven query-based protocol is needed to mitigate this waste of resources. Many papers propose a distributed approach to data-management in sensor networks [8], [18], [24]. Instead of having a central database collecting sensor readings from individual nodes, the data is stored and distributed over the nodes themselves. Queries then extract the necessary data from the sensor nodes.

In a query-based distributed scheme the node needs to be alert for any data queries. Thus, making the node awake (i.e., consuming energy) even if it is not sending or collecting data. A desirable property of WSNs is power-efficiency. The main objective of this work is to save power and prolong the life of nodes by making them consume as less power as possible. Having low sleep cycle times will make a node handle incoming requests faster, since they are only waiting for a short period. However, a longer sleeping cycle will minimize the number of switches between sleeping and active modes. These switches incur a communication overhead and therefore additional energy consumption; a newly awaken node must poll its parent node for requests. Furthermore, the waking time (the time required for a node to switch from sleeping to active) ranges between 2 to 13.2 milliseconds for XBEE series 2 nodes [4]. Thus, we aim to maximize sleeping time while we maintain QoS requirements to minimize the number of switches.

In this paper we use a queueing system model to capture delay characteristics. We study this model to explore its suitability in capturing our system. Specifically, since we are having a two-way communication dynamic with packets generations interdependent to each other, it is not intuitively clear whether we can find a suitable solution to model our system. Our contribution is threefold:

- We demonstrate the effect of maximizing the sleep cycle duration on energy consumption. The paper shows that even with the same cumulative sleep duration, the number of switches from active to sleep state incur a significant effect on energy.
- We explore a queueing system model's capability in capturing a query-based WSN system. In particular, we perform an experimental evaluation of MAC layer dynamics to aid in developing the model.
- We perform a simulation and experimental study of the system. Also, we demonstrate the validity of our findings.

The paper is divided as follows. Section 2 details the motivation of our research, Section 3 outlines related work, Section 4 summarizes the technical background of the prototype, Section 5 describes the system design of the prototype, Section 6 proposes a model for QoS in our system, Section 7 evaluates the model both with the prototype and with simulations, Section 8 concludes with a summary and directions for future work.

II. Motivation

In our project we tackle one of the main challenges in WSNs, namely energy efficiency. Sensors have a limited amount of power. In the absence of any tasks, a node wastes its energy waiting for transmissions. Making a node sleep (i.e., consume less power) in inactive periods is a well-studied way to mitigate energy waste [6]. However, we consider the

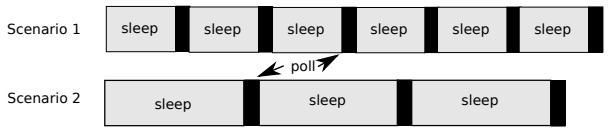

Fig. 1: A motivating scenario on the effect of sleep cycle time on energy consumption

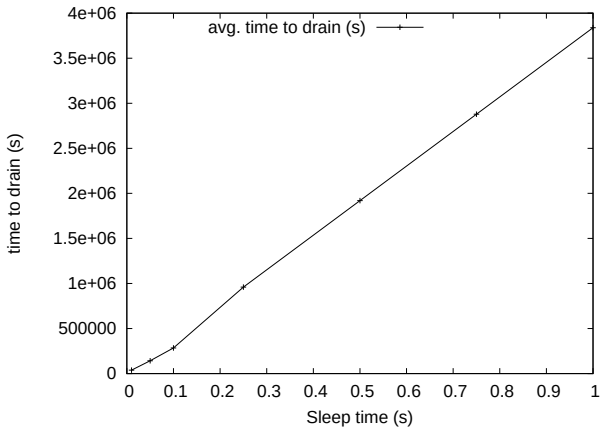

Fig. 2: Amount of time required to drain batteries for different sleep cycle times with no workload

effect of the sleep cycle duration on energy consumption. We show a motivating example in Figure 1. In it we consider two scenarios where no data exist to be received. The first scenario has a sleep cycle duration equal to half of that of the second scenario. The black rectangles denote the period of polling for buffered packets. It is apparent that scenario one consumes more energy since it is switching to poll for received packets more often. The effect of sleep cycle duration extends to scenarios with active receptions and can be intuitively derived. Figure 2 shows the effect of sleep cycle times on the longevity of nodes. It is shown that a sleep time of 1.0 seconds increase longevity by a factor of 10000% compared to 0.01 seconds.

Given our motivation, the main objective of our work is to maximize sleeping times. However, there is a trade-off between sleep times and QoS requirements (we consider delay); a longer sleep cycle duration translates to a larger delay times. A query-based framework is considered for our implementation [8]. We will use ZigBee, based on IEEE 802.15.4 [19], as our system's infrastructure. The objective of our work is threefold: First, we will deploy a query-based WSN using Arduino micro-controllers [1] and XBEE modules [3]. Second, we will implement the query-based scheme over QualNet simulator [2]. Using the deployment and simulation framework we can study the system for the effects of sleep cycle duration on QoS and energy consumption. Third, a mathematical model will be constructed to capture QoS and energy characteristics. From the model, a formula for determining maximum sleeping times is derived. The resulted formula will be tested in the deployment (as a proof-of-concept prototype) and in simulation (to investigate scalability).

The main contribution of our work is to produce a mathematical model that can be used to capture QoS with respect to sleep behavior in WSNs. This model can then be used to derive suitable sleeping times when given certain QoS characteristics and network load. A queueing system [14] can be used to model this problem. An M/G/1 queueing model with vacations [10] is, we claim, suitable for such a problem.

Another challenge is to deploy and simulate a query-based WSN. A full deployment of transaction handling distributively in WSNs would include transaction processing and optimization, routing, aggregation, etc. [8]. We will start with a simple query-based scheme in our deployments. However, we will design it so it would be possible to incrementally develop it to include more transactional features.

III. Related Work

A transactional framework for WSNs was proposed in [8], [17]. Our framework is based on them. Specifically we will use the notion of one-time and persistent queries. Queries specify rules that include among others: (1) the duration of needed observations. (2) frequency. (3) upper/lower thresholds. (4) desired locations/sensors. and (5) QoS requirements. In [8], a hybrid storage system to maintain historical data is proposed; historical data should be sent out by the node to a central site where it is stored for later retrieval. We pursue a hybrid approach where a user can query a node directly or through the front-end. If a central node or entity was interested in maintaining the history of a specific node, it can poll it on demand or issue a persistent query. A persistent query tells a node to send collected data for a specific duration and rate, e.g., send me the temperature every ten minutes for the next seven days.

Energy conservation gained a lot of attention in the WSNs community [6], [15], [16], [25]; proposals to design MAC and routing protocols, in addition to topology control that are energy aware are examples of such work. We, however, consider a more practical approach; we consider network topology is a given. This is important since we cannot assume that we have control on the topology or movement of nodes. Also, we consider off-the-shelf hardware. Thus, we model our solution around the IEEE 802.15.4 standard and AODV protocol [21]. Some other work also tried to model IEEE 802.15.4 networks [9], [11], [13], [18], [20], [22]. Some work even model the network for energy efficiency [9], [11], [22]. Our work focuses on on-demand query-based query-based WSNs. This type of WSNs has also been treated for energy efficiency consideration [12], [23]. In [12], the authors try to minimize energy consumption by sleeping. They develop a probabilistic model that trade-off between energy and data quality. Our model extends the work done in the literature of modeling query-based WSNs by the following. We consider the trade-off between QoS and energy efficiency. Also, we employ an M/G/1 model with vacations to capture the system and derive appropriate sleeping times given QoS requirements. Similar employment of this mathematical treatment is done in [5] for optical networks. We use their approach and modify it to model IEEE 802.15.4 sleeping times.

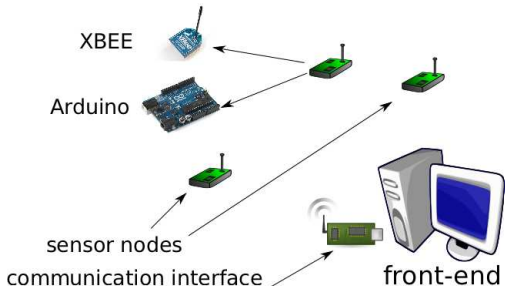

Fig. 3: Components used in the prototype

Sensor networks are being widely deployed for measurement, detection and surveillance applications. Related work to data processing in a wireless sensor network is TinyDB, a query processing system for extracting information from a sensor network. TinyDB provides a SQL like interface, for each input request it collects data from the nodes, filters it and sends the aggregated result to a PC via power efficient network processing algorithm. Queried data consists of sensor attributes (e.g., location) as well as sensor data. A typical monitoring scenario involves aggregate queries or correlation queries. In order to optimize the system's performance, result of these queries are partially aggregated at each intermediate node. In our system we have implemented an interface to query on the properties such as the sensor temperature and further grouped the results on the basis of floor numbers.

IV. Prototype infrastructure

A depiction of our prototype is supplied in Figure 3. It consists of a front-end that receives external requests and delivers readings for outside users. The back-end is a collection of sensor nodes consisting of a micro-controller and a wireless module. A detailed description of used components follows:

- *Arduino*: Arduino is a micro-controller capable of sensing the environment by receiving input from sensors in its surroundings. The micro-controller on the board is programmed using the Arduino programming language (based on C/C++ library) and the Arduino development environment (based on *Processing*). In Arduino the connectors are exposed in such a way that allows CPU boards to be connected to a variety of interchangeable add-on modules. In our experiments we use Arduino Uno boards.
- *XBEE*: XBee is a radio module based on IEEE 802.15.4 networking protocol providing wireless end-point connectivity to devices. It is designed for high-throughput applications which require low latency and predictable communication timing. AT commands are used to control the radio settings. They can operate either in a transparent data mode or in a packet based (API) mode. In the transparent mode data coming into the DIN (Data IN) pin gets directly transmitted to the intended receiving radios without any modification. Incoming packets can either be directly addressed to one target or broadcast to multiple targets. This mode is primarily used in instances where an existing protocol cannot tolerate changes to the data format. We use XBee series 2 modules.
- *ZigBee*: ZigBee is a specification for a suite of high level communication protocols using small, low-power digital radios based on an IEEE 802 standard for Personal Area Network (PAN). ZigBee builds upon the physical layer and medium access control defined in IEEE standard 802.15.4 (2003 version) for low-rate wireless PANs. It is targeted at radio-frequency (RF) applications that require a low data rate, long battery life, and secure networking. ZigBee data rate varies from 20 kbps to 900 kbps but 250 kbps is best suited for periodic or intermittent data or a single signal transmission from a sensor or input device. The current ZigBee protocol supports beacon and non-beacon enabled networks. In our experiments we operate in non-beacon mode. There are three different types of Zigbee devices:
  - *ZigBee Coordinator (ZC)*: it forms the root of the network tree and might bridge to other networks. There is exactly one ZC per network.
  - *ZigBee Router (ZR)*: it can run an application and can also act as an intermediate router.
  - *ZigBee End Device (ZED)*: it has the capability to talk to the parent node. It cannot relay data from other devices. This relationship allows the node to be asleep a significant amount of the time thereby giving long battery life.

V. System design

A. WSN as a DB

Wireless networks as virtual databases is an emerging area of research interest. The most popular data model in use today is the relational data model. We use the relational model in our front-end. SQL type interface is present which provides a way to query data using various SQL operators. The main goal of WSN as database is to preserve the location transparency. So the idea is to allow the user to generate the query on a sensor network without having knowledge of the topology. An example of such queries is: `SELECT AVG(temperature) FROM $all WHERE floor = '2'`, where the average temperature of nodes in the second floor is calculated.

B. Sleeping Scheme

We use a cyclic sleeping scheme. A node goes to sleep if a specific amount of time, we call it *idle_time*, passed with no activity. Activity includes transmissions and receptions to or from the node. The sleep cycle time is fixed. When a sleep cycle finishes, the node wakes up and pull its parent node. Upon polling, the parent node will transmit all buffered packets to the node. In our prototype and simulation we choose the value of idle_time to be very small to allow for longer sleep duration. Furthermore, under load that has a mean time of inter-arrivals less than *idle_time*, the node might not go to sleep. This can make the node active all the time and making us not benefiting from having an upper threshold of delay requirements. Thus, it is necessary to sleep almost immediately after serving packets.

C. Front-end design

The front-end design consists of a web based user interface listing the wireless sensor nodes present in the topology. This UI queries the data from a back-end MySQL database. There is a provision to drill down the report on the basis of attributes such as nodeId, floor number, etc. We have implemented a query processor to fetch the data from nodes based on the attributes such as their physical location (which is floor number in our case). The query interface will fetch the details of all nodes that satisfies the condition (*i.e.*, floor number in our case) and then send query request packets to them. The result of this query gets stored in the database and it can be used later to perform any analysis. So the role of query interface is to filter the nodes based on the query condition and then stores the result into the database. The front-end also includes a communication module that generate packets in the format acceptable by XBee nodes and wait for replies from sensors.

D. Simulation Design

The simulations are done in Qualnet 5.1. All simulations consist of one PAN coordinator node and a specific number of end nodes (Reduced Function Devices). All nodes are part of the same wireless sub-net. All nodes operate using the 802.15.4 wireless standard. Routing is done in a static fashion. The beacon order of all nodes is set to 15. This disables the beaconing mode of the mesh network. Therefore, the nodes operate by polling the coordinator for new data with a preset polling interval. In between poll cycles, the transmission interfaces of the end nodes can go to sleep.

The unmodified version of Qualnet does not allow end nodes to sleep in a non-beacon-mode 802.15.4 mesh network. In order to implement this behavior, we had to change the source code of Qualnet. Specifically this involved commenting out `|| sscs802_15_4->t_BO == 15` on line 379 in the file *sscs_802_15_4.cpp* of the sensor networks library.

Traffic generation in our simulations is accomplished by using a `SUPER-APP` traffic generator for each end node. The source of the traffic is always the coordinator. All network communication is done unreliable (UDP). The size of the request package is 8 bytes. The package frequency is exponentially distributed. The mean of the package frequency as well as the number of packets is determined based on the desired workload of the overall network. The reply function of the traffic generator is set to reply with exactly one packet, that also has a payload of eight bytes. These replies are send out with a delay of exactly 100ms.

VI. Modeling wait time

In this section, we will develop a model of delay and queue utilization of our system. The Round Trip Time (RTT) experienced by each packet is described by the following sum:

$$\psi = W + W_{endnode} \tag{1}$$

where W is the wait time in the front-end, $W_{endnode}$ is the wait time experienced by the end-node. We focus on the W since it is effected by the dynamics of sleeping and buffering, where $W_{endnode}$ is predictable and can be substituted with a constant value dependent on the channel characteristics and processing requirements. We will use an M/G/1 queue model with vacations. We denote the amount of incoming traffic to the network by λ. The distribution of inter-arrival times, $A(t)$, is exponential, where the mean is $\frac{1}{\lambda}$. The service time, denoted as X, is translated as the time required to service the packet in the head of the queue of the front-end. This value is dependent on the contention of the medium.

For such a system, the wait time can be calculated by the following equation:

$$W = \frac{\lambda \bar{X^2}}{2(1-\rho)} + \frac{\bar{V^2}}{2\bar{V}} \tag{2}$$

where ρ is the intensity and is equal to $\lambda\bar{X}$, and V is the random process describing wait times. We assume that V is discrete. In this relation, $\bar{X^2}$ is the only unknown. The second expectation is given as the sum of variance and squared mean. As we reviewed in the related work section, many solutions were proposed to solve the problem of finding the service time for CSMA/CA protocols. However, we face a novel problem for finding the service time of our framework that was not tackled for sensor networks to the best of the authors knowledge. In our framework, the sleeping agent is desperate from the queueing agent. Furthermore, the service time include the transmission from the front-end which occur in batches, processing and queueing in the back-end, and finally transmitting the requested value back to the front end. This introduces complexity into figuring out the service time as a closed-form solution and it remains as an open question worthy of investigation. We take a step into this investigation and take an unconventional mean into determining the service time, hence an experimental approach. We will provide our approach, methodology, and findings in the evaluation section. In it we propose an experimental approximation methodology and prove its validity through experiments.

VII. Evaluation

In this section we will present an evaluation of our framework using simulation and a prototype implementation. We use QualNet [2] with a sensor networks plug-in for simulations. Each simulation run takes 15 minutes. The reported workloads (λ or l in the figures) are the number of requests per second for the whole network. A node is selected uniformly as a destination of each request. Prototype results are averaged over 1000 requests. Reported results are for end-to-end delay of requests from front-end to end-node unless mentioned otherwise. Parts used for our prototype are over-viewed in Section IV. The evaluation will start with analyzing the behavior of service times. Then, delay time measurements are displayed and compared with the model. After that, we show results for the prototype implementation. Finally, we will explore the efficacy of using the model to maximize sleeping times while maintaining an average delay.

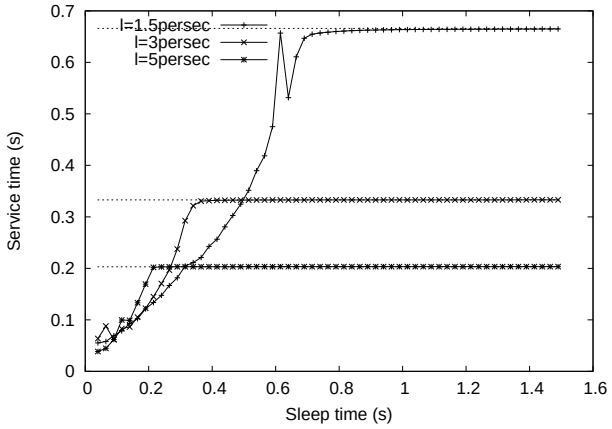

Fig. 4: Service time as calculated from the model

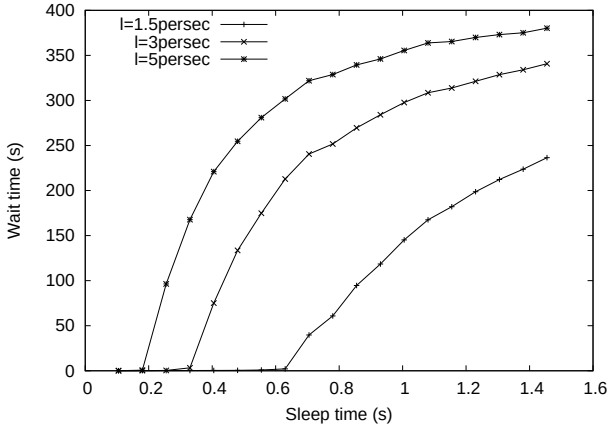

Fig. 5: Wait times of a three-node network

A. Service time

In this section we will investigate the service time, mentioned in section VI. We will observe two topologies with one and two end-nodes. In each experiment, we will plot the average service time while varying the sleep time for different workloads. The calculated service time is obtained by rearranging Equation 2 and solving for it by substituting the remaining known variables. We make an assumption that the distribution of service times is discrete; results in next subsections support that this is a working approximation.

In Figure 4 the service times of a topology with three end-nodes are displayed while varying sleeping times. The figure shows results for three different workloads. It is clear from the figure that the service time is upper bounded. The interesting observation is that this upper bound is equal to the inverse of the workload, *i.e.*, the average inter-arrival time. This confirms our hypothesis since the intensity of the system is not allowed to exceed a value of one. We call the point where the service time converges as the *saturation point*. Prior to the saturation point, the service time increases proportionally to sleep time and workload. This observation can help us in using our model; for sleeping times smaller than the saturation point we set the value for the service time according to

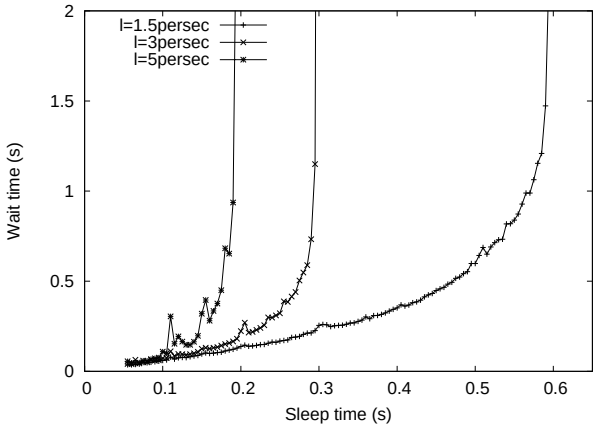

Fig. 6: Wait times of a three-node network focusing on non-saturated channel values

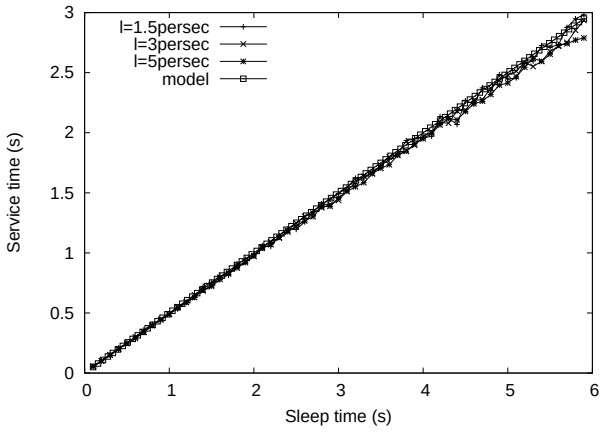

Fig. 7: Sleep times for a network with one end-node compared with the model

an approximation of a monotonic function. We use a linear approximation that will guarantee an upper bound of service time. Otherwise, sleeping time is set to be approaching the average inter-arrival time.

Another observation is that although the service time values are increasing prior to saturation, their effect is less observable until the intensity is approaching 1. This is due the dominance of the second term in equation 2 that corresponds to (*sleeping time*). This is demonstrated in Figures 5 and 6. Prior to the saturation point, the wait times are relatively small and dominated by the sleeping time, as shown in Figure 6. However, after the saturation point, Figure 5, the intensity is approaching 1 and therefore causes a dramatic increase in wait times.

B. Model validation

In this section we will demonstrate the behavior of 1 and 2 end-nodes scenarios. We first observe how a small value of service time can make the wait time derivable from the sleeping time only. First, we show a baseline case for a topology with one end-node in Figure 7. In this scenario, the channel does not saturate and the service times are always

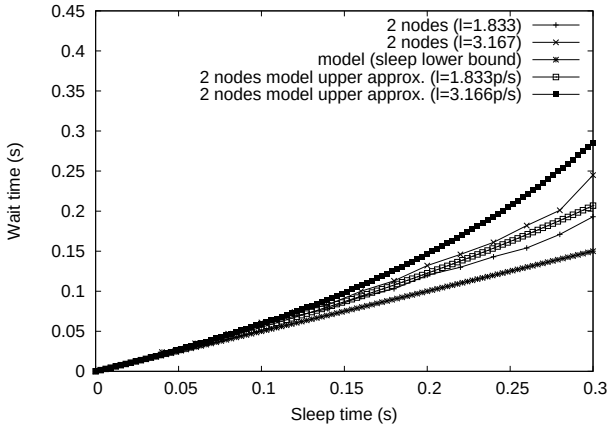

Fig. 8: Comparing wait times for a 2-node network with upper and lower bounds obtained through model

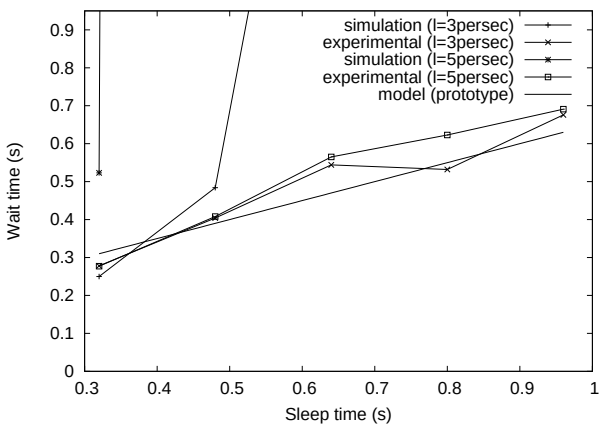

Fig. 10: Comparing wait times of simulation, prototype, and model for a two-node network

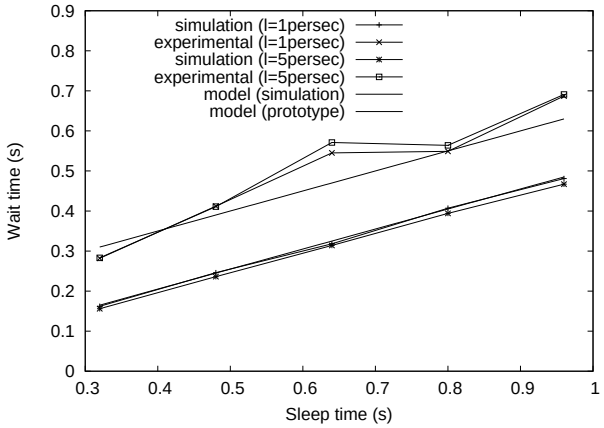

Fig. 9: Comparing wait times of simulation, prototype, and model for a single node network

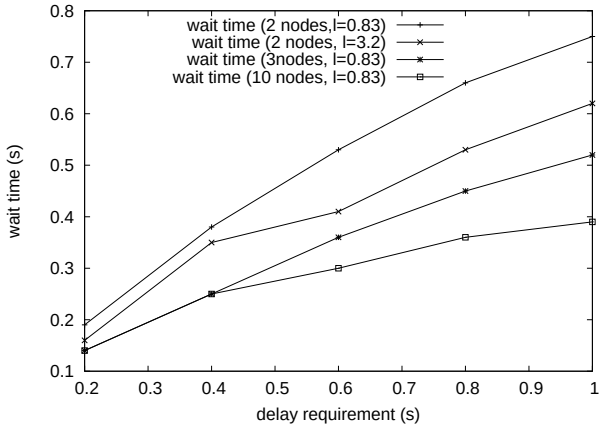

Fig. 11: assigning sleep times to achieve an upper bound on wait times for networks with 2 and 3 nodes

small, rendering wait times to be dominated by sleep times. Therefore, the difference in workloads is not observed and all plots overlap with the model. Now, we show a scenario with two end-nodes and show how the model predicts behavior when sleep times approaches saturation point. Figure 8 shows these results. Two runs with different workloads are shown. A lower bound is the delay experienced by the sleep cycle and is experienced regardless of workload. An upper bound is calculated using our approximation. We observe that as the number of nodes or amount of workload is increasing, the approximation becomes more conservative.

C. Prototype Evaluation

Now we will display results obtained from our model to demonstrate its behavior in accordance to our model. In Figure 9 the results of a single-node prototype is compared to simulation and model. Results shown for the prototype are of the RTT due to difficulty in measuring end-to-end request delay. The same behavior is experienced by both node. However, Since we include the whole RTT for the prototype the effect of end-node transmission and processing is accounted, which is approximated to be equal to 0.1 seconds. Next, we present results for a two-node network in Figure 10. It is observed that the prototype acts similarly to the single-node network case, hence saturation is not achieved. However, the simulation saturated and values became much larger than the expected non-saturated case. This demonstrated the effect of channel and physical condition on the saturation point. We were not able to test for larger number of nodes in our prototype due to the lack of hardware. We however, showed that our findings for the saturated case are observed in the prototype.

D. Assigning sleeping times

In this section we will put our findings to the test. For three networks with 2, 3 and 10 nodes respectively, we will calculate sleeping times that guarantee an upper bound for delay. Results are shown in Figure 11. Having a larger number of nodes or a larger workload make the approximation more conservative, as shown in the figure. Furthermore, a larger requirement make the calculated sleeping time more conservative relatively. This is due approaching saturation which makes the calculation of

wait times more sensitive to service times.

VIII. Conclusions

In this paper we studied a query-based WSN where nodes are requested for information on-demand. Specifically, we tackle the problem of energy efficiency. We demonstrated that sleep times have a significant impact on energy consumption even with the same cumulative sleeping time. We model the system as a M/G/1 queueing model to capture wait times with respect to workload, sleep time, and MAC layer characteristics. A simulation study is performed to aid and validate our findings. Furthermore, a prototype implementation of the framework was presented. Finally, We showed that we can maintain upper-bound wait times for delay using our model.

This work can aid in developing closed-form models to model MAC layer dynamics for the considered framework. Furthermore, we are currently exploring the possibility of maintaining a strict upper-bound on wait times using the model. This can be achieved by limiting the number of requests on fly. This limit can be achieved by a simple derivation from the presented model.

References

[1] Arduino. http://www.arduino.cc.

[2] Scalable networks technologies. http://www.scalable-networks.com.

[3] Xbee. http://www.digi.com/xbee.

[4] *XBee RF Module – Product Manual*, v1.xex edition. http://ftp1.digi.com/support/documentation/90000982_B.pdf.

[5] P. H. Ho Ahmad R. Dhaini and G. Shen. Energy efficiency in ethernet passive optical networks for how long can onu sleep. In *In Proceedings of IEEE GLOBECOM11*, 2011.

[6] Giuseppe Anastasi, Marco Conti, Mario Di Francesco, and Andrea Passarella. Energy conservation in wireless sensor networks: A survey. *Ad Hoc Networks*, 7(3):537 – 568, 2009.

[7] Luigi Atzori, Antonio Iera, and Giacomo Morabito. The internet of things: A survey. *Computer Networks*, 54(15):2787 – 2805, 2010.

[8] Philippe Bonnet, Johannes Gehrke, and Praveen Seshadri. Towards sensor database systems. In Kian-Lee Tan, Michael Franklin, and John Lui, editors, *Mobile Data Management*, volume 1987 of *Lecture Notes in Computer Science*, pages 3–14. Springer Berlin / Heidelberg, 2001. 10.1007/3-540-44498-X_1.

[9] B. Bougard, F. Catthoor, D.C. Daly, A. Chandrakasan, and W. Dehaene. Energy efficiency of the ieee 802.15.4 standard in dense wireless microsensor networks: modeling and improvement perspectives. In *Design, Automation and Test in Europe, 2005. Proceedings*, pages 196 – 201 Vol. 1, march 2005.

[10] B. T. Doshi. Queueing systems with vacations — a survey. *Queueing Systems*, 1:29–66, 1986. 10.1007/BF01149327.

[11] Rick W. Ha, Pin-Han Ho, and Xuemin Sherman Shen. Optimal sleep scheduling with transmission range assignment in application-specific wireless sensor networks. *International Journal of Sensor Networks*, 1(1-2):72–88, 2006.

[12] Q. Han, Sharad Mehrotra, and Nalini Venkatasubramanian. Energy efficient data collection in distributed sensor environments. In *Distributed Computing Systems, 2004. Proceedings. 24th International Conference on*, pages 590 – 597, 2004.

[13] C.Y. Jung, H.Y. Hwang, D.K. Sung, and G.U. Hwang. Enhanced markov chain model and throughput analysis of the slotted csma/ca for ieee 802.15.4 under unsaturated traffic conditions. *Vehicular Technology, IEEE Transactions on*, 58(1):473 –478, jan. 2009.

[14] Leonard Kleinrock. *Theory, Volume 1, Queueing Systems*. Wiley-Interscience, 1975.

[15] Dilip Kumar, Trilok C. Aseri, and R.B. Patel. Eehc: Energy efficient heterogeneous clustered scheme for wireless sensor networks. *Computer Communications*, 32(4):662 – 667, 2009.

[16] Zhiwu Liu and Wei Wu. A dynamic multi-radio multi-channel mac protocol for wireless sensor networks. In *Proceedings of the 2010 Second International Conference on Communication Software and Networks*, ICCSN '10, pages 105–109, Washington, DC, USA, 2010. IEEE Computer Society.

[17] Sam Madden, Joe Hellerstein, and Wei Hong. *TinyDB: In-Network Query Processing in TinyOS*, 2003.

[18] A. Manjeshwar, Qing-An Zeng, and D.P. Agrawal. An analytical model for information retrieval in wireless sensor networks using enhanced apteen protocol. *Parallel and Distributed Systems, IEEE Transactions on*, 13(12):1290 – 1302, dec 2002.

[19] S. Middleton. Ieee 802.15 wpan low rate study group par. Technical Report doc. number IEEE P802.15-00/248r3, IEEE, 2002.

[20] E. D. N. Ndih, N. Khaled, and G. De Micheli. An Analytical Model for the Contention Access Period of the Slotted IEEE 802.15.4 with Service Differentiation. In *Proceedings of the IEEE International Conference on Communications (ICC)*, pages 1–6, 2009.

[21] Charles E. Perkins, Elizabeth M. Belding-Royer, and Samir R. Das. Ad hoc on-demand distance vector (aodv) routing. RFC Experimental 3561, Internet Engineering Task Force, July 2003.

[22] Qingchun Ren and Qilian Liang. Energy-efficient medium access control protocols for wireless sensor networks. *EURASIP J. Wirel. Commun. Netw.*, 2006:5–5, April 2006.

[23] Mohamed A. Sharaf, Jonathan Beaver, Alexandros Labrinidis, and Panos K. Chrysanthis. Balancing energy efficiency and quality of aggregate data in sensor networks. *The VLDB Journal*, 13:384–403, 2004.

[24] Yong Yao and Johannes Gehrke. The cougar approach to in-network query processing in sensor networks. *SIGMOD Rec.*, 31:9–18, September 2002.

[25] Wei Ye, J. Heidemann, and D. Estrin. An energy-efficient mac protocol for wireless sensor networks. In *INFOCOM 2002. Twenty-First Annual Joint Conference of the IEEE Computer and Communications Societies. Proceedings. IEEE*, volume 3, pages 1567 – 1576 vol.3, 2002.